CMT and Coding Conventions.

Brett Viren

Physics Department

BROOKHAVEN
NATIONAL LABORATORY

Daya Bay Collaboration Meeting, Aug. 2006

Why do we need to bother with conventions?

We need to follow a set of conventions:

- CMT needs to know how to interpret a file:
  - Is it a library source file?
  - Is it a public or a private header file?
  - Does it provide the `main()` function?
  - Is it a ROOT `LinkDef.h` file?
- Humans need a common language and organization or we go crazy!

BROOKHAVEN NATIONAL LABORATORY

What does CMT do?

We use CMT for:

- Building our software in a (mostly) platform independent way.
- Managing environment variables (PATH, LD_LIBRARY_PATH, etc).
- Hooking in external packages.

What we don't use CMT for, but could (and probably eventually should):

- Defining and managing software releases.

BROOKHAVEN NATIONAL LABORATORY

Overview of code organization

Our code should be split in to *Packages*

- Mostly independent and never circularly dependent. Dependencies include:
  - Compile time dependencies due to `#include<>`
  - Run time dependencies due to dynamic linking
- A package includes:
  - At most one library with a set of public header files
  - At most one main application program
  - Test application(s) (or library if we adopt a unit-testing framework)
  - Examples and documentation

BROOKHAVEN NATIONAL LABORATORY

Libraries

`Package/src/` holds source files (`.cc`) that make up a library and any private header files (`.hh`) that are **not** needed by other packages to use the library.

`Package/include/` public header files that **are** required for other packages to use this library.

BROOKHAVEN NATIONAL LABORATORY

Applications

`Package/app/` holds a single source file that defines a `main()` function. Possible to support multiple applications in one Package, but not yet.

`Package/src/` holds any additional application implementation files. These may also be used to form a library instead of linking them all directly into the application. If used for a library, Library conventions apply

BROOKHAVEN NATIONAL LABORATORY

Misc

`Package/cmt/requirements` (next slide)

`Package/doc/` holds any and all documentation beyond what is included inline with Doxygen.

`Package/example/` holds any files that provide example use of the package (eg. Root or Geant4 scripts).

`Package/support/` holds files that the package needs to work but are not part of the build. Examples:

- Root scripts (`.C` files) that are needed to drive some the package's application.
- Root data (`.root` files) providing some default input.
- Geant4 "macros" (`.mac` files) that set up standard running conditions.

`Package/xml/` holds G4dyb XML files (only for G4dyb package, new packages should place XML in `Package/support/`)

BROOKHAVEN NATIONAL LABORATORY

The Package/cmt/requirements file

This file tells CMT what to do with the package. It is usually very short and contains these kind of things:

Declare the package name:

```
package G4dyb
```

List the dependencies on other packages:

```
use CLHEP "*"  External  # Specify:
use GEANT "*" External   #   1)Package name
use XERCES "*" External  #   2)Version ("*" means any)
use MCEvent "*" G4dyb    #   3)Project name
```

Use one or more "pattern" to tell CMT what to do:

```
# For applications
apply_pattern monolithic_application
# For a library's public headers
apply_pattern install_includes_auto
# If ROOTCINT dictionaries need generation
apply_pattern rootcint_dictionary
# Build a shared library
apply_pattern shared_library
```

BROOKHAVEN NATIONAL LABORATORY